RFC: Dataset Object Header Size

Vailin Choi
John Mainzer
Jacob (Jake) Smith

The document discusses enhancement to the HDF5 library to minimize object header size when a dataset is created, and describes our proposed implementation of this enhancement. The feature minimizes the metadata to raw data ratio when an application creates large numbers of tiny datasets.

Introduction

For one of The HDF Group projects, statistics were gathered from Axom test data files, generated by a run of MFEM's ex9p.cpp application. The statistics show the amount of space used for the objects' metadata and also total metadata space allocated. It was noted that there was a high percentage of unused space in dataset object headers -- 204 bytes on average. The amount of unused space for groups was negligible.

The HDF5 library uses a hard-coded constant for the dataset's initial object header size even though not all space is used. Reducing initial dataset object header size to the point that it is only large enough for the required messages should reduce the metadata to raw data ratio significantly for applications (such as Axom) which create many tiny datasets – albeit at the cost of inefficiencies, should additional attributes be attached to the datasets. This RFC describes our proposed plan for implementing this change.

As the HDF5 library has no way of knowing if the user will attach attributes to a dataset after it is created, we propose to minimize the dataset object header size only when directed by the user. We do this for two reasons:

- Reduction in dataset object header size is irrelevant unless the dataset is tiny, and
- Minimization of object header size will result in immediate allocation of an object header continuation block if an attribute is added to a dataset whose object header size has been minimized.

Current implementation

When creating a dataset in an HDF5 file, the library will first allocate memory for the *H5D_t* data structure and will also set up info for the following messages for a dataset:

1) Dataspace message
2) Datatype message
3) Fill value message
4) Layout message
5) Filter pipeline message if layout is chunked and filter is used
6) External file list message if external file list is defined
7) Modification time message if the object header is tracking time and the version is 1

Then it will call *H5O_create()* with the parameter *size_hint* to set up the dataset's object header:

- Allocate memory for the object header *H5O_t* and initialize information in the data structure
- Allocate file space via *H5MF_alloc()* for the object header prefix plus *size_hint*, which is the estimated size for the first chunk of object header messages. File space will be allocated out of this chunk later on for the required messages.
- Allocate memory for the object header prefix and the first chunk of object header messages

The callers for *H5O_create()* can be:

- Group
- Dataset
- Committed datatype
- Superblock

The *size_hint* parameter passed to *H5O_create()* for a dataset is a set define:

```
/* Set the minimum object header size to create objects with */
#define H5D_MINHDR_SIZE 256
```

This is the estimated size for eight possible messages when a dataset is created:

```
#define H5O_NMESGS  8       /* initial number of messages        */
```

The eight possible messages are the seven messages listed above plus the object header continuation message. While the estimate is reasonable and efficient for a general use case, it can cause significant file space inefficiencies when an application (such as Axom) creates large numbers of tiny datasets. This RFC addresses the issue of the *size_hint* passed to *H5O_create()* for a dataset.

Additions to the API

We will introduce two sets of public routines for users to set the minimum object header size for a dataset.
The first set *H5Pset/get_dset_no_attrs_hint()* allows the user to set the minimization request on a per dataset basis. A new property H5D_CRT_MIN_DSET_HDR_SIZE_NAME will be introduced in the dataset creation property list.
The second set *H5Fset/get_dset_no_attrs_hint ()* allows the user to set the minimization request on a per file basis. Users can set/unset the request for the creation of datasets while the file is opened.

H5Pset_ *dset_no_attrs_hint*

The new API will be:

```
herr_t H5Pset_dset_no_attrs_hint(hid_t dcpl_id, hbool_t minimize)
```

The parameters are:

- *dcpl_id*: the dataset creation property list
- *minimize*: TRUE or FALSE in using the minimum object header size when creating the dataset; the library default is FALSE

H5Pget_*dset_no_attrs_hint*

The new API will be:

```
herr_t H5Pget_dset_no_attrs_hint(hid_t dcpl_id, hbool_t *minimize)
```

The parameters are:

- *dcpl_id*: the dataset creation property list
- *minimize*: whether the minimum object header size is used or not when creating the dataset

H5Fset_*dset_no_attrs_hint*

The new API will be:

```
herr_t H5Fset_dset_no_attrs_hint(hid_t file_id, hbool_t minimize)
```

The parameters are:

- *file_id*: the file identifier
- *status*: TRUE or FALSE in using the minimum object header size when creating datasets in the file; the library default is FALSE

H5Fget_*dset_no_attrs_hint*

The new API will be:

```
herr_t H5Fget_dset_no_attrs_hint(hid_t file_id, hbool_t *minimize)
```

The parameters are:

- *file_id*: the file identifier
- *minimize*: whether the minimum object header size is used or not when creating datasets in the file

Library Internal Changes

When creating a dataset in the file, the library calls *H5O_create()* with the *size_hint* parameter to set up an object header for the dataset. To determine the size for *size_hint*, the library will first access the H5D_CRT_MIN_DSET_HDR_SIZE_NAME property from the dataset's creation property list (DCPL). It will then decide whether to use the minimum object header size for the dataset as follows:

- If the property does not exist, the action to minimize the object header size will depend on the file-enabled *minimize* status.
- If the property does exist, the library will retrieve the *minimize* status from the DCPL. The action to minimize the object header size will depend on the dcpl-enabled or the file-enabled *minimize* status.

The table below lists the actions to be taken.

Property exists	*dcpl-enabled* `minimize`	*file-enabled* `minimize`	*Action*
No	--	True	Use minimization
No	--	False	Use library default
Yes	True	True	Use minimization
Yes	True	False	Use minimization
Yes	False	True	Use minimization
Yes	False	False	Use library default

To summarize: the minimized object header size will be used if either DCPL property or file setting is TRUE, and any TRUE setting will override a FALSE (or undefined) in the other mode.
If the action is to use the library default, the size H5D_MINHDR_SIZE of 256 will be passed in *size_hint*.
If the action is to use minimization, the library will determine the minimum object header size, which will be passed in *size_hint*, by summing up the size of messages required[1] for the dataset at creation. It will call the existing internal routine *H5O_msg_size_oh()* to get the size for the following messages:

- Dataspace message with H5O_SDSPACE_ID
- Datatype message with H5O_DTYPE_ID
- Fill value message with H5O_FILL_NEW_ID
    - It may also create the old fill value message with H5O_FILL_ID for backward compatibility

[1] Which is typically a proper subset of the 8 possible messages listed in Section 2 above.

- Layout message with H5O_LAYOUT_ID
- Filter pipeline message with H5O_PLINE_ID
    - If layout is chunked and filter is used
- External file list message with H5O_EFL_ID
    - If external file list is defined
- Modification time message with H5O_MTIME_NEW_ID
    - If the object header is tracking time (H5O_HDR_STORE_TIMES)
    - If the object header is version 1
- Object header continuation message with H5O_CONT_ID

Since the object header version may be needed to determine the size or inclusion of some of the above messages, we will split the coding in *H5O_create()* into three parts:

- Part I: Create an object header in memory and assign its version.
    - *H5O__create_ohdr()*
- Part II: Calculate required space for the object header.
- Part III: Allocate space (including space for compact datasets), complete header initialization, and apply the header to the file.
    - *H5O__apply_ohdr()*

New internal functions will be created as necessary to carry out these operations. This procedure will be implemented in such a way that the public API and other types of object header creation (groups, e.g.) are unaffected.

Implementation Details

The file setting for dataset header minimization is in a shared structure, so multiple file IDs referring to the same underlying file will all have the most recent setting made at the file level with *H5Fset_dset_no_attrs_hint()*.

The existing API call to instantiate a new object header, *H5O_create()*, will be retained, as it is used in multiple places throughout the library – reducing the amount of changes to be made – and provides a valuable layer of abstraction. However, as we need a way to override the *size_hint* variable it expects, and to calculate the minimum required space before allocating space in the file for the object header, two new functions are created at the internal level, *H5O__create_ohdr()* – responsible for creating a header object and assigning bare-minimum setup (such as version) – and *H5O__apply_ohdr()* – responsible for allocating space in the file and complete initialization, both of which *H5O_create()* will call in order, passing the size in *size_hint* to *H5O__apply_ohdr()* as the size to allocate.

When using a minimized dataset, these new functions –*H5O__create_ohdr()* and *H5O__apply_ohdr()* – will be called directly, with the intervening step of using the created object header to calculate the size to be allocated. This intermediary step is static to *H5Dint.c*, *H5D__calculate_minimum_header_size()*.

In *H5O__apply_ohdr()*, the received size is adjusted as necessary, to align bytes to expected boundaries (if applicable), and – through the macro *H5O_SIZEOF_HDR()* – add space for a compact dataset.

Fortran/C++/Java

Update Fortran, C++, and Java wrappers for the four new public routines.

Testing

Add tests to *test/dsets.*c for *H5Pset/get_dset_no_attrs_hint* and add tests to *test/tfile.c* for *H5Fset/get_dset_no_attrs_hint:*

- Verify these public routines set and get the *minimize* status correctly
- Enable minimum object header size for datasets via *H5Fset_dset_no_attrs_hint() or H5Pset_dset_no_attrs_hint();* verify the object header size for the dataset created in the file is the amount expected.
- Verify that a dataset whose object header size has been minimized handles the addition of user attributes correctly (albeit not efficiently).
- Interleave the creation of datasets with minimum object header size enabled/disabled via dcpl and/or the file; verify the object header size for datasets is the amount expected

Requisite additions to *test/gen_plist* and *test/enc_dec_plist* will be made, to facilitate property list testing.

Add tests in *test/ohdr*, *test/tattr*, and *test/tsohm* to verify appropriate behavior of minimized dataset object headers in various situations and with different demands made on them.

Documentation

- Generate reference manual entries for the four new public routines

Other consideration

There is another suggested fix to resolve this jira issue via the existing public routine *H5Pset_attr_phase_change(ocpl_id, max_compact, min_dense).* This routine sets the threshold values, *max_compact* and *min_dense*, for attribute storage. These thresholds determine the point at which attribute storage changes from compact

storage (i.e. storage in the object header) to dense storage (i.e., storage in a heap and indexed with a B-tree). The default for *max_compact* is 8 and *min_dense* is 6.
The suggested fix is of two-fold:

- The internal library will calculate the minimum object header size at dataset creation as the total of the following:
    - Size for the dataset's required messages like dataspace, datatype etc.
    - Size of attributes: *max_compact* * *H5D_ATTRHDR_SIZE* (a new constant define probably set to 32 bytes)
- User application will set *max_compact* to 0 via this public routine indicating to the library that no attributes are expected for the object

This suggested fix has the benefit of no changes to the current API. However, further investigation indicates the following ramifications:

- The setting via this public routine is associated with the latest format
- Attributes will be stored densely regardless by setting *max_compact* to 0
- Obscure the original purpose of this public routine
- For the default case when *max_compact* is 8, the above calculation for the minimum object header size will be greater than the current constant define H5D_MINHDR_SIZE

Revision History

May 12, 2018	Version 0 – Initial draft
May 15, 2018	Version 1 – after John's review and edits
May 18, 2018	Version 2 – Update for Fortran, C++, and Java wrappers
June 20, 2018	Version 3—Add H5Fset/get_minimize_dset_hdr_size
June 28 2018	Version 4 – Renamed H5F/Pget/set_minimize_dset_hdr_size to H5F/Pget/set_*dset_no_attrs_hint*
Dec. 28 2018	Version 5 – Add implementation details and amend testing details